VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MOBILNÍ APLIKACE PRO PLÁNOVÁNÍ DOSAŽENÍ CÍLE

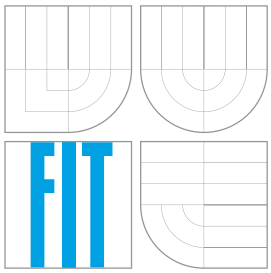
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VLADIMÍR BOBULA

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MOBILNÍ APLIKACE PRO PLÁNOVÁNÍ DOSAŽENÍ CÍLE

MOBILE APPLICATION FOR GOAL PLANNING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — VLADIMÍR BOBULA
AUTHOR

VEDOUCÍ PRÁCE — Ing. IGOR SZŐKE, Ph.D.
SUPERVISOR

BRNO 2014

Abstrakt

Cílem této práce je navrhnout a implementovat mobilní aplikaci pro plánování dosažení cíle. Aplikace je určena pro systém Android. V práci jsou popsány základy systému Android. Následně je zanalyzován návrh grafického uživatelského rozhraní a databázy se snahou využít techniky gamifikace. V implementační části jsou vysvětleny části aplikace spolu s řešením nalezených problémů. Poslední část obsahuje popis testování aplikace a její zveřejnění službou Google Play.

Abstract

The aim of this thesis is to design and implement a mobile application for goal planning. The application is designed for the Android system. The paper describes the basics of this system. Afterwards, the design of graphic user interface and database with the effort to use gamification is analysed. The implementation section explains parts of application as well as solutions of occurred problems. The last section contains a description of testing the application and its following publication via Google Play.

Klíčová slova

Mobilní, aplikace, plánování, cíl, časová os, Android, Google Play

Keywords

Mobile, application, plan, goal, timeline, Android, Google Play

Citace

Vladimír Bobula: Mobilní aplikace pro plánování dosažení cíle, bakalářská práce, Brno, FIT VUT v Brně, 2014

Mobilní aplikace pro plánování dosažení cíle

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Igora Szökeho, Ph.D. a uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Vladimír Bobula
21. května 2014

Poděkování

Ďakujem vedúcemu práce Ing. Igorovi Szökemu, Ph.D. za venovaný čas, konzultácie, odborné rady a vedenie.

Obsah

Kapitola 1

Úvod

Úlohou tejto práce bolo vytvoriť mobilnú aplikáciu, poskytujúcu rozvrhnutie plánu k dosiahnutiu cieľa, stanovenie úloh a sledovanie plnenia plánu. Aplikácia bola vyvinutá pre systém Android. Pri vývoji práce som sa zameral najmä na plánovanie postupu pri riešení diplomových prác. Hlavnou cieľovou skupinou sú teda študenti vysokých škôl. Preto sa podieľali na vývoji aplikácie jej testovaním.

Tento text popisuje vývoj mobilnej aplikácie od návrhu až po publikovanie v službe Google Play.

Čitateľovi bude predstavená platforma Android, základné komponenty, prvky, ktoré boli v aplikácii použité.

Kapitola popisujúca návrh aplikácie obsahuje porovnanie aplikácií na podobný účel, návrh grafického používateľského rozhrania, databázy a pridanie gamifikácie do aplikácie.

V kapitole obsahujúcej implementáciu, je popis implementácie aplikácie rozdelený na časti podľa účelu. Riešenie správy projektov, časových osí, úloh, zobrazenia prehľadu postupu v dosahovaní cieľov je bližšie špecifikované v tejto kapitole. Metódy rozhraní fragmentov, obsluha prvku ActionBar a vysúvacieho menu sú objasnené v popise aktivity aplikácie. Kapitolu uzatvára popis práce s databázou.

V poslednej časti sú popísané postupy pri testovaní, zámery testovania a postup pridávania a upravovania častí aplikácie podľa získanej spätnej väzby používateľov.

V závere sú zhrnuté výsledky práce a zhodnotenie splnenie cieľa. Aplikácia je posúdená z hľadiska konkurencieschopnosti, je jej vymedzený budúci účel použitia, cieľová skupina používateľov a sú navrhnuté ďalšie možnosti vývoja.

Kapitola 2

Vývoj mobilnej aplikácie pre systém Android

Na začiatku tejto kapitoly sú uvedené dôvody pre vývoj mobilnej aplikácie a výber platformy. Následne je popísaný softvér použitý pri vývoji tejto práce. Zvyšok kapitoly je o systéme Android.

2.1 Dôvody pre vývoj aplikácií pre Android

V súčasnosti je malé percento ľudí, ktorí nevlastnia mobilný telefón. S príchodom inteligentných telefónov (smartfónov) sa rozširujú možnosti použitia telefónov. Okrem pôvodného účelu pribúdajú mnohé pokročilé možnosti ako napríklad prístup k internetu (cez mobilné alebo Wi-Fi siete), GPS navigačná jednotka, rôzne senzory, veľké množstvo aplikácií. Každým rokom sa zvyšuje počet používateľov smartfónov.

Väčšina moderných smartfónov obsahuje dotykový displej, hardvérová klávesnica je skôr na ústupe. Smartfón je postavený na určitom operačnom systéme. Medzi najznámejšie patrí Android od firmy Google, iOS od firmy Apple, Windows Phone od firmy Microsoft, Symbian, BlackBerry 10, Bada, WebOS.

Podľa analýzy spoločnosti IDC mal Android v štvrtom štvrťroku roku 2014 78-percentný podiel na trhu smartfónov[1].

Zvyšovaním výkonu a možností využitia inteligentných mobilných telefónov dochádza k tomu, že v určitých prípadoch dokážu plnohodnotne nahradiť počítač. Mobilné telefóny majú ľudia pri sebe a tým sa zvyšuje potenciál použitia mobilných aplikácií.

2.2 Príprava prostiedkov pre vývoj aplikácie

Väčšina aplikácií pre Android je programovaných v jazyku Java. Systém Android pracuje s balíkmi Application package file (APK). V nich je bajtkód Dalvik, ktorý vznikne prekladom zdrojových kódov v jazyku Java. V tejto práci som programoval v jazyku Java, preto som najskôr nainštaloval vývojovú sadu jazyka Java – Java development kit (JDK) – dostupú na stránkach firmy Oracle[2].

[1] http://www.idc.com/getdoc.jsp?containerId=prUS24676414

[2] http://www.oracle.com/technetwork/java/javase/downloads/index.html

Knižnice s rozhraním pre programovanie aplikácií a vývojárske nástroje potrebné na zostavenie, testovanie a ladenie aplikácií som získal nainštalovaním sady Android SDK cez nástroj Android SDK Manager.

Dôležitým vývojovým nástrojom je emulátor Android zariadení. Vďaka nemu je možné vyvíjať aplikácie bez fyzického zariadenia a testovať aplikácie na rôznych virtuálnych zariadeniach (Android Virtual Device – AVD). AVD je možné vytvoriť v nástroji AVD Manager. Pri vytváraní AVD sú dostupné nastavenia zariadenia, ako napríklad verzia systému Android, hardvérové nastavenia. Pre uľahčenie vývoja aplikácie som použil integrované vývojové prostredie Eclipse. Hlavným dôvodom tohto výberu bola priama podpora tohto vývojového prostredia Androidom vo forme zásuvného modulu Android Development Tools (ADT).

2.3 Základy programovania pre Android

Aplikácie tvorí kombinácia rôznych komponentov. Komponent môže spustiť iný komponent použitím intentu. Operačný systém Android je postavený na Linuxovom jadre [4]. Je viacužívateľský, pričom v roli používateľov sú jednotlivé aplikácie. Každá aplikácia má podľa základného nastavenia spustený vlastný proces a priradené unikátne používateľské číslo. Android spustí proces pri potrebe vykonávať nejakú časť aplikácie. Proces je ukončený, ak už nie je ďalej užitočný alebo v dôsledku uvoľňovania pamäte systémom pre jej nedostatok. Kódy aplikácií sa vykonávajú izolovane, pretože každý proces má vlastný virtuálny stroj (virtual machine).

2.3.1 Manifest aplikácie

Každá aplikácia obsahuje v koreňovom priečinku projektu súbor `AndroidManifest.xml`[3]. V manifeste sú základné informácie o aplikácii. Systém môže spustiť komponent iba ak je deklarovaný v manifeste.

Okrem iného manifest obsahuje:

- Názov Java balíku aplikácie.
- Deklarácie komponentov, z ktorých je aplikácia zložená. Na základe uvedených informácií o komponentoch systém vyhodnotí, za akých podmienok môže byť komponent spustený.
- Deklarácie povolení aplikácie k prístupu chránených častí aplikačného rozhrania a povolení, ktoré musia mať ostatné aplikácie k prístupu k danej aplikácii.
- Deklaráciu minimálnej verzie systému, ktorú aplikácia vyžaduje.

2.3.2 Komponenty Android aplikácie

Komponenty sú základnými prvkami aplikácií. Pomáhajú definovať aplikáciu. Každý komponent je unikátna časť, cez ktorú môže systém pristupovať k aplikácii. Niektoré komponenty na sebe môžu byť závislé. Komponenty delíme podľa účelu na štyri rôzne typy. Každý typ má odlišný životný cyklus.

[3] http://developer.android.com/guide/topics/manifest/manifest-intro.html

Aktivity

Entita systému reprezentujúca obsah jednej obrazovky s používateľským rozhraním [5]. Väčšina aplikácií má viac aktivít. Tvoria celok, ale sú na sebe nezávislé. Pre systém Android je typické časté otváranie a zatváranie (napr. tlačidlo späť) nových aktivít. Aktivita môže spustiť aktivitu inej aplikácie.

Služby

Komponenty navrhnuté k neustálej prevádzke[4]. Bežia na pozadí. Neposkytujú používateľské rozhranie. Vykonávajú prácu pre riadiace procesy. Iný komponent môže spustiť službu a komunikovať s ňou.

Poskytovatelia obsahu

Poskytujú úroveň abstrakcie pre prácu s dátami aplikácie[5]. Definujú prístup k dátam. Sú štandardným rozhraním pre poskytnutie dát jedného procesu iným procesom. K prístupu dátam cez poskytovateľa obsahu sa pošle požiadavka o dáta, poskytovateľ vykoná požadovanú akciu a vráti výsledok.

Zámery

Slúžia ako správy rozposielané v systéme pre indikáciu zmeny a aplikácie na ne môžu reagovať[6]. Nezobrazujú užívateľské rozhranie. V prípade výskytu určitej udalosti môžu zobraziť notifikáciu pre upozornenie používateľa. Zámery môžu okrem systému (napr. upozornenie na nízky stav batérie) vytvárať aj aplikácie a spúšťať pomocou nich iné aktivity.

2.3.3 Spúšťanie aktivít

Aplikácie je vhodné deliť do aktivít (napríklad podľa logických celkov). Pre spúštanie aktivít sa používajú zámery.[6] Najskôr je nutné vytvoriť zámer. Ten môže byť explicitný alebo implicitný. Explicitný určuje konkrétny komponent, ktorý systém ihneď spúšťa. Implicitný deklaruje akciu, ale vyhodnotenie, ktorý komponent sa spustí, je ponechané na systém. Systém vyhľadáva aktivitu porovnávaním obsahu zámeru s filtrami zámerov, umiestnených v manifestoch iných aplikácií na danom zariadení. V prípade zhody systém spúšťa komponent a predá mu zámer. V prípade zhody viacerých filtrov systém zobrazí voľbu a výber komponentu je ponechaný na používateľovi. Do zámeru vložíme názov komponentu (explicitný) alebo akciu a dáta (implicitný). Akcia je reťazec, ktorý určuje aká akcia sa vykoná. Je možné vytvoriť vlastné akcie alebo použiť vstavané. Dáta slúžia k popisu dát spojených so zámerom a to pomocou URI (objekt typu uri) alebo typom dát MIME alebo ich kombináciou. Typ MIME je taktiež možný použiť vlastný ako aj vstavaný. Samotné spustenie aktivity je vykonané metódou startActivity, do ktorej je zámer vložený ako argument. V prípade potreby výsledku spustenia aktivity je možné použiť StartActivityForResult.

[4] http://developer.android.com/guide/components/services.html
[5] http://developer.android.com/guide/topics/providers/content-providers.html
[6] http://developer.android.com/guide/components/intents-filters.html

2.3.4 Používateľské rozhranie

Všetko, čo používateľ vidí a s čím môže pracovať je používateľské rozhranie aplikácie. Android poskytuje veľa komponentov, z ktorých je možné poskladať rozhranie. Ako predpis pre rozvrhnuie týchto komponentov slúži layout. Všetky elementy rozhrania aplikácie sú odvodené od triedy View[7]. Špecifickým typom View je podtrieda ViewGroup, ktorá umožňuje zoskupovať a usporiadať triedy odvodené od View. Použitím týchto tried vzniká stromová hierarchia.

Layout

Definuje štruktúru používateľského rozhrania[8]. Layout je možné vytvoriť deklarovaním elementov rozhrania v súbore formátu XML alebo vytvárať elementy vytváraním inštancií tried View. Android umožňuje vytváranie a spravovanie rozhrania obidvomi spôsobmi. Typicky sa to využíva tak, že rozloženie je deklarované v XML súbore, ale v zdrojovom kóde sa nastavuje stav jeho objektov.

Deklarovaním XML layoutu sa oddelí vzhľad aplikácie od jej chovania definovaného v zdrojovom kóde. Toto umožňuje zmeniť vzhľad bez nutnosti zásahu do kódu [3]. Pre konkrétne situácie (napr. rôzne rozlíšenia, orientácie obrazoviek a odlišné jazyky) môžu byť vytvorené špecifické layouty. Nastavenie zobrazenia layoutu určíme v aplikačnom kóde aktivity.

Pre deklarovanie elementov používateľského rozhrania v XML súboroch platí, že na začiatku súboru je uvedená XML hlavička. Každý súbor s layoutom musí obsahovať koreňový element. Tým môže byť objekt typu View alebo ViewGroup, ktorý môže obsahovať ďalšie objekty. Názvy elementov a ich vlastností sú pomenované podľa štruktúry a názvov tried a metód. Pre XML formát trieda predstavuje názov elementu a triedne metódy zodpovedajú názvom atribútov. Pre jednoznačnú identifikáciu objektov View, môžu objekty obsahovať atribút ID. Layout je nutné uložiť v súbore s príponou .xml v priečinku projektu `res/layout`.

Kontajnery zabaľujú synovské prvky a organizujú ich do rôznych štruktúr. Sú odvodené od triedy ViewGroup. Najpoužívanejšie typy kontajnerov:

- Lineárny kontajner – Organizuje synovské prvky lineárne podľa zvolenej orientácie. V prípade horizontálnej orientácie do riadku, v prípade vertikálnej do stĺpca. Ak výška obsahu presiahne výšku layoutu, vytvorí sa posuvník. V tomto druhu kontajnera môže byť synovskému prvku priradený atribút weight, ktorý určuje, koľko miesta má prvok zabrať vzhľadom k ostatným prvkom.

- Relatívny kontajner – Pre usporiadanie synovských prvkov používa relatívne pozície. Každý prvok môže byť umiestnený relatívne k iným prvkom (súrodencom) alebo ku kontajneru (rodičovskému prvku).

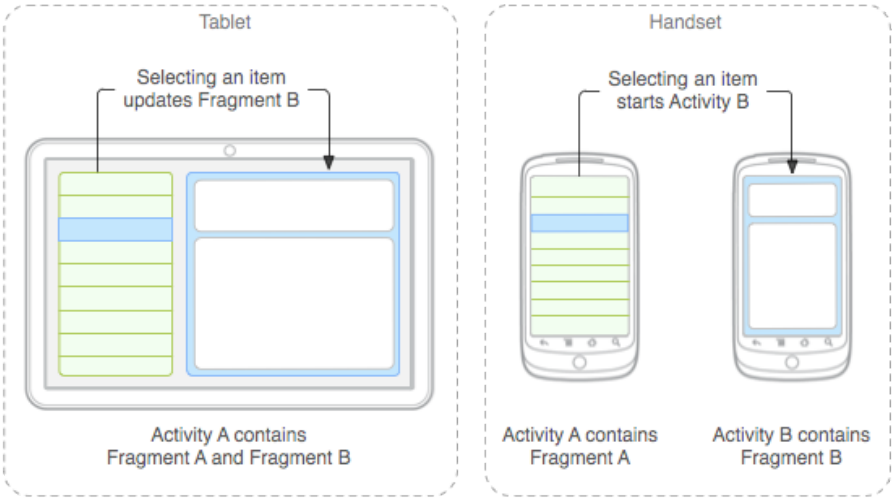
- Tabuľkový kontajner – Usporiadanie prvkov do riadkov a stĺpcov. Riadok môže obsahovať nula alebo viac buniek, pričom každá bunka môže obsahovať jeden objekt triedy View.

[7] http://developer.android.com/guide/topics/ui/overview.html
[8] http://developer.android.com/guide/topics/ui/declaring-layout.html

2.3.5 Fragmenty

Fragmenty vznikli v Androide až v aplikačnom rozhraní verzie 11. Pôvodne boli určené pre tablety, aby pomohli riešiť komplikácie vznikajúce pri rôznych veľkostiach obrazoviek. Možnosti, ktoré poskytujú, sú však výhodné pre použitie aj v mobilných telefónoch. Fragment je medzivrstva medzi aktivitou a používateľským rozhraním.

Fragment môže obsahovať chovanie alebo časť používateľského rozhrania a vytvoriť tak znovupoužiteľné moduly. Musí byť vždy vložený do nejakej aktivity[9]. Aktivita môže používať viac fragmentov súčasne a fragment môže byť znovu použitý vo viacerých aktivitách. Jeho životný cyklus je priamo ovplyvnený životným cyklom aktivity, ktorá ho používa.



Obrázek 2.1: Príklad použitia dvoch fragmentov v jednej aktivite. Na tablete sú zobrazené obidva fragmenty, na mobilnom telefóne iba jeden.[9]

Fragment môže byť súčasťou layoutu aktivity. V tom prípade je priradený k ViewGroup v hierarchii rozloženia aktivity. Vkladá sa deklarovaním do súboru layoutu xml elementom fragment alebo použitím triedy FragmentTransaction v zdrojovom kóde aktivity. Fragment má v tomto druhu použitia vlastný layout. Je možné fragment použiť aj na vykonávanie nejakej činnosti na pozadí a v tom prípade nemá vlastné používateľské rozhranie.

2.3.6 Prostriedky

Aplikácie obsahujú prostiedky – všetko spojené s vizuálnym zobrazením aplikácie (reťazce, obrázky, hudobné súbory, ponuky, animácie, farby, štýly, grafické rozvrnutie aplikácie, ...). Je doporučené ukladať všetky prostriedky v priečinku `/res` umiestnenom v koreňovom priečinku[10]. Tým, že sú prostriedky oddelené od zdrojového kódu, je možné ich zmeniť na jednom mieste bez nutnosti meniť kód. Ďalším dôležitým dôvodom je používanie rôznych prostriedkov pre rôzne zariadenia (napríklad jazykové mutácie, veľkosť a orientácia displeja). Prostriedky sa delia na štandardné a alternatívne. Štandardné sú použité, pokiaľ neexistujú alternatívne, ktoré by zodpovedali použitému zariadeniu. Rozlíšenie alternatívnych prostriedkov umožňujú kvalifikátory. Kvalifikátor je krátky reťazec, ktorý je možné pridať

[9] http://developer.android.com/guide/components/fragments.html
[10] http://developer.android.com/guide/topics/resources/overview.html

do názvu priečinka. Popisuje nastavenie zariadenia, pre ktoré budú použité prostriedky v tomto priečinku. Android podporuje veľa odlišných kvalifikátorov.

2.3.7 Možnosti ukladania

Android poskytuje rôzne možnosti pre ukladanie trvalých dát aplikácií. Dôležitými aspektmi pre výber spôsobu ukladania dát je množstvo priestoru, ktoré budú potrebovať a prístup k nim[11].

- Zdieľané nastavenia (Shared Preferences) – Pre ukladanie súkromných dát (nie iba nastavení) ako primitívnych dátových typov v pároch kľúč – hodnota. Rozhranie pre tento spôsob uloženia poskytuje trieda SharedPreferences.
- Vnútorné úložisko – Uloženie súkromných dát v pamäti zariadenia. Prednastavene k takto uloženým dátam iné aplikácie nemajú prístup. Po odinštalovaný aplikácie sú tieto dáta odstránené.
- Vonkajšie úložisko – Uloženie verejných dát v zdieľanej vonkajšej pamäti (napríklad SD karta).
- SQLite databáza – Uloženie štruktúrovaných dát v súkromnej databáze.
- Sieťový server – Uloženie dát na webe s vlastným sieťovým serverom.

Používanie databázy

Android má plnú podporu pre databázu SQLite. Všetky databázy sú prístupné zadaním názvu iba triedam aplikácie. Názvy databáz musia byť unikátne v rámci aplikácie. Doporučený spôsob vytvárania novej SQLite databázy je vytvoriť podtriedu triedy SQLiteOpenHelper a prepísať metódu onCreate. V tejto metóde je možné vytvoriť tabuľky príkazmi SQLite.

S databázou sa pracuje cez jej inštanciu. Metóda `getReadableDatabase` slúži pre získanie databáze pre čítanie. Pre zapisovanie do databáze je metóda `getWritableDatabase`. Obidve vracajú objekt triedy SQLiteDatabase reprezentujúci databázu. Ten má triedne metódy pre vykonanie SQLite operácií.

2.3.8 Správca obsahu

Správca obsahu je voliteľný komponent (trieda ContentProvider)[12]. Poskytuje prístup pre čítanie a zapisovanie dát aplikácie iným aplikáciám. Poskytuje mechanizmy pre definovanie zabezpečenia dát.

Aplikácia pristupuje k dátam správcu obsahu cez klientský objekt triedy ContentResolver. Ten poskytuje metódy pre prácu s dátami. Metódy sú nazvané rovnako ako v inštancii správcu obsahu (objekt konkrétnej podtriedy ContentProvider).

2.3.9 Ďalšie využité triedy

Niektoré z využitých tried pri implementovaní aplikácie.

[11] http://developer.android.com/guide/topics/data/data-storage.html
[12] http://developer.android.com/guide/topics/providers/content-provider-creating.html

ActionBar

ActionBar je súčasť okna v hornej časti obrazovky[13]. Poskytuje možnosť zobrazenia názvu a ikony aplikácie, aktuálnej pozícii používateľa v aplikácii a zobrazenia jeho možností. Použitím triedy ActionBar je dosiahnuté podobné rozhranie v rôznych aplikáciách priblížením sa k najnovšiemu rozhraniu systému Android [1]. ActionBar bol do systému Android pridaný až vo verzii 3.0. Aby bolo možné používať ActionBar v starších verziách, bol pridaný do knižnice Support Library od verzie systému 7 – Android 2.1. Knižnica má názov appcompat a musí byť pre zaistenie kompatibility priložená do projektu aplikácie. Pri používaní appcompat má ActionBar niektoré metódy rozhrania odlišné.

Obrázek 2.2: Zobrazenie prvku ActionBar. 1 - ikona aplikácie, 2 - položky pre akcie, 3 - položka pre zobrazenie ďalších akcií.[13]

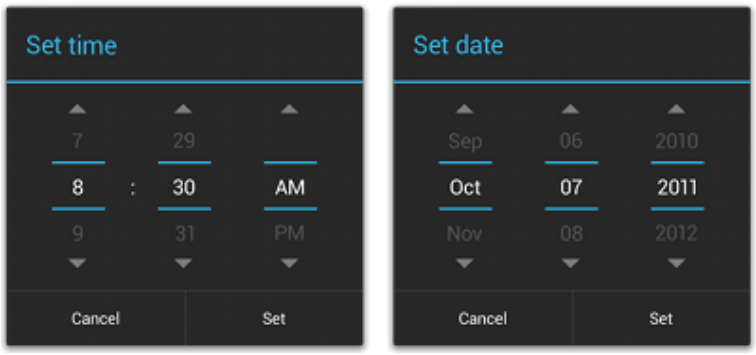
Vysúvacie menu

Poskytuje navigáciu v aplikácii. Menu sa vysunie po ťahu prstom od ľavej časti obrazovky smerom do stredu alebo po stlačení ikony v prvku ActionBar[14]. Vďaka knižnici Support Library je možné používať ho v systémoch od verzie 4. V aplikačnom rozhraní ho reprezentuje trieda DrawerLayout. Pre použitie tohto menu sa v súbore s rozložením používateľského rozhrania deklaruje ako koreňový element prvok DrawerLayout, ktorého potomkami by mal byť prvok pre zobrazenie obsahu aplikácie počas skrytého menu a prvok zobrazujúci obsah menu.

Výber dátumu a času

Pre výber dátumu a času Android ponúka hotové riešenie v triedach TimePickerDialog a DatePickerDialog. Sú to dialógy, ktoré poskytujú používateľské rozhranie pre vloženie hodnôt[15]. Okrem toho zaisťujú kontrolu správnosti zadaného vstupu. Dialógy je možné nastaviť konštruktorom. Zmenu dátumu, času, je možné nastaviť metódami updateDate, updateTime. Rozhranie OnDateSetListener obsahuje abstraktnú metódu onDateSet pre oznámenie o zadaní hodnôt používateľom a ich odovzdanie. Dokumentácia doporučuje používať pre sprostredkovanie dialógov triedu DialogFragment. Tá poskytuje rozhranie pre zobrazenie a správu dialógu.

[13] http://developer.android.com/guide/topics/ui/actionbar.html
[14] http://developer.android.com/design/patterns/navigation-drawer.html
[15] http://developer.android.com/guide/topics/ui/controls/pickers.html



Obrázek 2.3: Zobrazenie dialógov pre výber dátumu a času.[15]

2.4 Podpora verzií platformy Android

Vývojom platformy dochádza k deleniu na verzie. To so sebou prináša komplikácie. Nová verzia môže obsahovať nové aplikačné rozhranie, ktoré predošlé verzie nepodporujú. Veľmi žiadané je podporovať čo možno najviac aktívnych zariadení. Kvôli možnosti využiť nové rozhrania a zároveň zachovať spätnú kompatibilitu verzií platformy, vývojári vytvorili sadu knižníc – Android Support Library. Pred samotnou implementáciou je vhodné určiť, ktoré verzie platformy majú byť s aplikáciou kompatibilné a následne tomu prispôsobiť implementáciu.

Kapitola 3

Návrh aplikácie

Pri návrhu aplikácie som sa snažil vytýčiť si ciele, ako napríklad hlavný zámer aplikácie a cieľová skupina používateľov. Vypracoval som si prehľad podobných služieb, aby som sa mohol inšpirovať, ale aj zistiť, čo im chýba. Vďaka tomu som získal predstavu, čo by aplikácia mala obsahovať a čomu sa naopak vyhnúť.

3.1 Zameranie aplikácie

Hlavným účelom mobilnej aplikácie je definovanie a sledovanie dosiahnutých cieľov. Aplikácia má využitie pre manažovanie projektov. Do projektu sú pridávané podúlohy, ktoré je možné po vykonaní označiť ako splnené.

Hlavnou cieľovou skupinou sú študenti, najmä vysokoškolskí. U nich je predpoklad, že riešia veľa projektov a aplikácia by pre nich mohla byť užitočná. Aplikácia je primárne určená pre plánovanie a sledovanie postupu na diplomových prácach.

3.2 Podobné služby

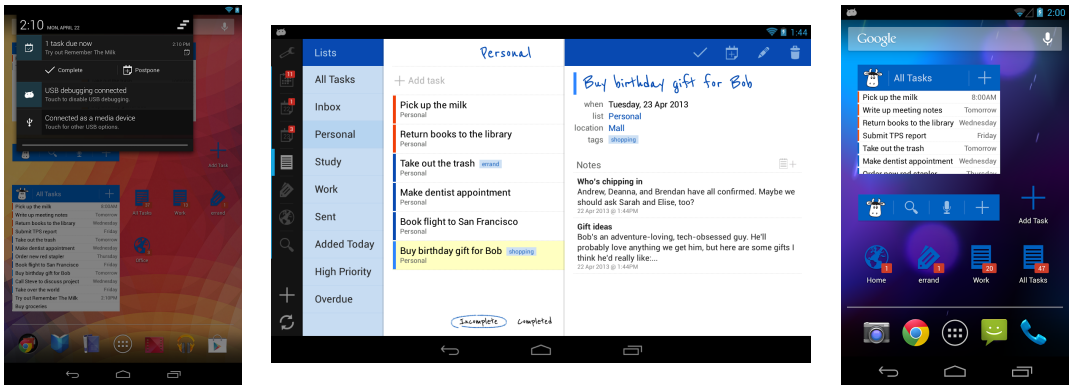
Pri vyhľadávaní podobných služieb som zistil, že je veľa podobných aplikácií. Zameral som sa na tie najpoužívanejšie. Zdrojom bola služba Google Play.

3.2.1 Remember The Milk

Android aplikácia Remember The Milk bola vytvorená firmou Remember The Milk Inc. a má viac ako 1 000 000 inštalácií. Okrem aplikácii pre Android existujú aj verzie aplikácie pre zariadenia značiek Apple a BlackBerry. Aplikácia vznikla ako rozšírenie ich existujúcej webovej aplikácie. Aplikácia má pekný dizajn a jednoduché ovládanie. Veľkou výhodou je možná integrácia do iných služieb, ako napríklad Gmail, Google calendar, Microsoft Outlook, Twitter, Evernote. Nevýhodou je, že funkcie ako neobmedzené automatické synchronizovanie s online službou, synchronizácia s viacerými zariadeniami, upozornenie na udalosti, miniaplikácie, sú spoplatnené a taktiež to, že úlohy nie je možné zdieľať.

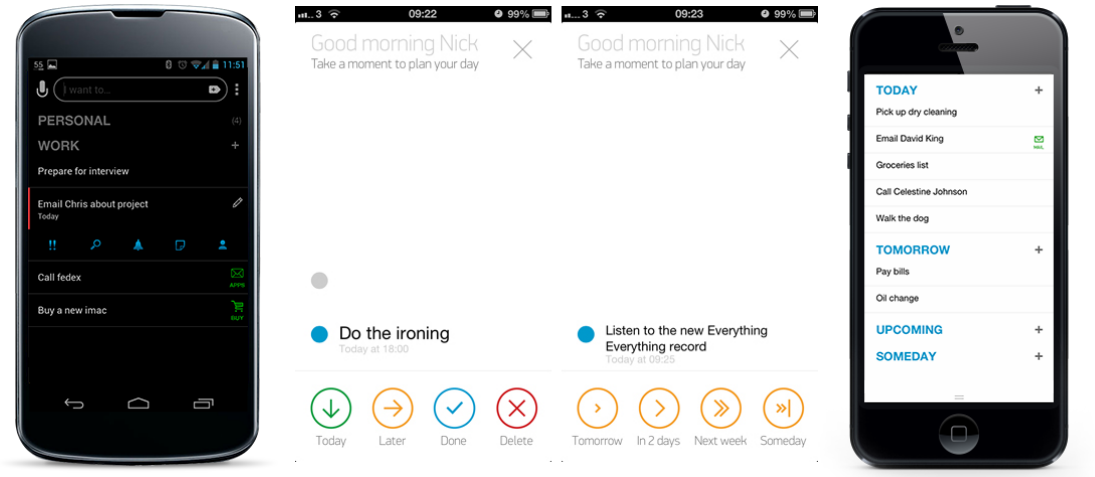
3.2.2 Any.DO To Do

Spoločnosť Any.DO vznikla v roku 2010. Vytvorila okrem iného Android aplikáciu Any.DO To Do, ktorá má v súčasnosti cez 5 000 000 inštalácií. Je dostupná na Android, aj zariadenia firmy Apple. Medzi jej výhody patrí vzhľad, intuitívne ovládanie, možnosť využiť pre



Obrázek 3.1: Na obrázkoch je vidieť rôzne funkcie aplikácie. Vľavo je vidieť notifikácie, vstrede spustenú aplikáciu, vpravo miniaplikácie.[2]

vytvorenie úloh mikrofón (automatický prevod slov do textu), pripomínanie úloh, no najmä možnosť zdieľania úloh. Aplikácia poskytuje synchronizáciu pre rôzne zariadenia v rámci danej aplikácie (a aplikácie Cal, ktorá je tiež od tejto firmy), formou rozšírenia pre webový prehliadač Chrome je možná aj pre Gmail. Aplikácia je zadarmo. Hlavnou nevýhodou je absencia synchronizácie s ostatnými službami (výrobca uvádza, že synchronizácia s mnohými službami je v pláne).

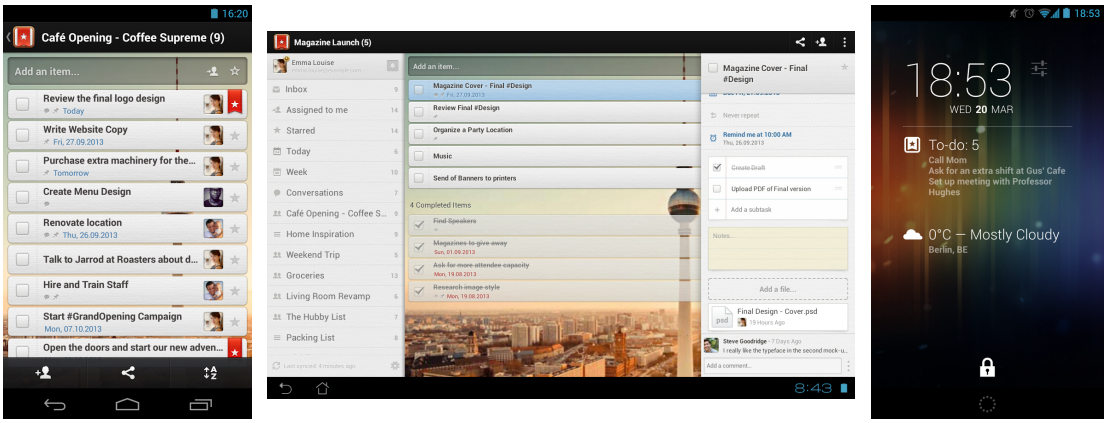


Obrázek 3.2: Obrázky zobrazujúce aplikáciu Any.DO To Do. Vľavo aplikácia pre Android, vpravo pre iPhone, v strede je zobrazené plánovanie úloh.[4]

3.2.3 Wunderlist – To-do & Task List

Startup 6Wunderkinder vytvoril v roku 2010 aplikáciu Wunderlist. Tá je dostupná pre zariadenia firmy Apple, pre operačný systém Microsoft Windows, pre Android (v súčasnosti cez 1 000 000 inštalácií). Aplikácia sa tak ako vyššie spomenuté aplikácie zaoberá manažovaním úloh a poskytuje väčšinu už spomenutých funkcií. Je možné používať platenú verziu aplikácie, ktorá umožňuje pridávanie súborov, priradenie úloh, komentáre a iné. Za nedostatok

považujem nemožnosť synchronizácie s inými službami.



Obrázek 3.3: Obrázky zobrazujúce aplikáciu Wunderlist.[5]

3.3 Postup návrhu GUI

Správne používateľské rozhranie by malo obsahovať vhodne navrhnuté mechanizmy pre vstup a výstup tak, aby uspokojilo používateľove potreby, možnosti a obmedzenia čo najefektívnejšie [2].

Po analýze riešení aplikácií s podobným zámerom som zistil, čo by mala aplikácia obsahovať a ako by malo plánovanie vyzerať. Aplikácia je určená na plánovanie dosiahnutie cieľa. Cieľ bude reprezentovať projekt. Plánovanie bude určené pre projekty trvajúce dlhšie. Primárne je aplikácia zameraná na plánovanie riešenia diplomových prác študentov. Projekt má časové ohraničenie a pre jeho splnenie musia byť splnené jeho úlohy. Projekt má časovú os, na ktorej sú rozložené jeho úlohy. Diskutovaním s potenciálnymi používateľmi som sa dozvedel, že v niektorých prípadoch by bolo vhodné rozdeliť projekt na viac častí. A to v prípade, že používateľ chce v rámci projektu spraviť nejakú jeho časť, ale chce prácu na nej oddeliť od hlavného riešenia projektu. Rozdelenie projektu je umožnené pridaním časových osí k projektu. Každá časová os projektu môže obsahovať úlohy.

V tejto verzii grafického používateľského rozhrania bola aplikácia testovaná používateľmi. Po spustení aplikácie sa zobrazil zoznam s projektmi a tlačidlo pre vytvorenie nového projektu. Po kliknutí na projekt sa spustila nová aktivita s údajmi projektu a zoznamom jeho osí. Zoznamy a vytváranie nových položiek boli implementované aktivitami. Pri vypĺňaní dátumov, v aktivite pre vytvorenie novej osi, môže používateľ využiť tlačidlo s ikonou hodín, ktoré zobrazí dialóg triedy DatePickerDialog pre výber dátumu. Možnosti navigácie v aplikácii boli veľmi obmedzené. K úlohám sa dalo dostať iba cez klikania na zoznamy a naspäť k projektom sa dalo dostať iba softvérovým alebo hardvérovým tlačidlom pre vrátenie späť. Najviac pripomienok používateľov súviselo s absenciou menu a tým, že úlohy neboli zoskupené podľa dátumov, čo pri viac úlohách bolo neprehľadné. Ďalšími poznámkami bolo, že ovládanie by bolo jednoduchšie a prehľadnejšie, keby bol použitý prvok ActionBar ako u mnohých súčasných aplikácií. Na základe týchto podnetov bola aplikácia prerobená. Používateľ si môže pri vytváraní osí zvoliť po akom časovom úseku bude časová os rozdelená. Automaticky sa vytvoria časové úseky, ktoré sú zobrazené po vybratí osi. Úseky sú zobrazené ako dátumy ohraničujúce trvanie úseku. Pri každom úseku je tlačidlo pre pridanie úlohy

Obrázek 3.4: Návrh prvotného GUI aplikácie.

k danému úseku. Neskôr bolo k úseku pridané aj hodnotenie prvkom RatingBar, ktorým môže používateľ poznamenať svoju spokojnosť s plnením úloh v úseku. Potom bol pridaný prvok ActionBar a vysúvacie menu – prvok DrawerLayout. Kvôli vysúvaciemu menu bola aplikácia prerobená na fragmenty a iba jednu aktivitu. Pri vybratí kategórie z tohto menu je zobrazený príslušný fragment. Fragmenty si pri vytváraní upravia názov a položky prvku ActionBar.

Obrázek 3.5: Ďalší návrh GUI aplikácie.

3.4 Návrh databázy

Aplikácia musí uchovať dáta používateľa aj po vypnutí aplikácie. Dáta majú medzi sebou väzby, preto je vhodné použiť relačnú databázu. Hlavným dôvodom pre použitie databázy SQLite je, že je súčasťou systému Android. Schéma databázy sa s vývojom aplikácie menila.

Hlavné časti však boli už v prvej verzii. Tabuľky boli určené podľa toho s akými dátami sa pracovalo v návrhu. Najskôr to boli tabuľky pre projekty, časové osi a úlohy. Neskôr bola pridaná tabuľka pre časové úseky. Každá tabuľka má stĺpec pre identifikátor zázamu. Tento je primárnym kľúčom tabuľky a je teda v rámci tabuľky jedinečný.

- Projekty – Obsahujú stĺpce pre textové atribúty projektov, názov a popis.
- Časové osi – Obsahujú stĺpce pre názov a číselné atribúty označujúce začiatok, koniec trvania a cudzí kľúč, odkazujúci do tabuľky projektov na projekt, ku ktorému záznam patrí.
- Časové úseky – Obsahujú stĺpce pre názov, začiatok, koniec trvania, hodnotenie a cudzí kľúč do tabuľky časových osí, určujúci os, ku ktorej úsek patrí. Do názvu budú vložené ohraničujúce dátumy. Hodnotenie slúži pre ukladanie spokojnosti používateľa s výkonom v danom úseku.
- Úlohy – Obsahujú stĺpce pre názov, popis úlohy, cudzí kľúč odkazujúci na časový úsek, ku ktorému patrí daná úloha a stĺpec pre pravdivostnú hodnotu označujúcu splnenie úlohy.

3.5 Gamifikácia

Gamifikácia je proces herného myslenia a herných mechanizmov pre zapojenie používateľov a riešenie problémov [6].

Pre lepšie zapojenie používateľov som pridal hodnotenie pomocou prvku RatingBar. Používateľ môže ohodnotiť spokojnosť so svojím výkonom v aktuálnom týždni.

Pre väčšiu motiváciu používateľa som do zobrazenia časovej osi pridal prvok ProgressBar, zobrazujúci jej priebeh. Ovládací prvok ProgressBar indikuje, ako blízko je používateľ k ukončeniu časovej osi.

Rozhodol som sa vytvoriť prehľad, v ktorom má používateľ možnosť vidieť svoju úspešnosť v percentách. Údaj je zobrazený farbou podľa úspešnosti.

Ďalšie zvýšenie iniciatívy používateľa by mohlo byť dosiahnuté pridaním bodového systému, zdieľaním postupu a výsledkov s ostatnými používateľmi, zavedením systému odmien.

Kapitola 4

Implementácia aplikácie

4.1 Projekty

V tejto časti sú popísané triedy implementujúce zobrazovanie a správu projektov.

4.1.1 Zoznam projektov – ProjectListFragment

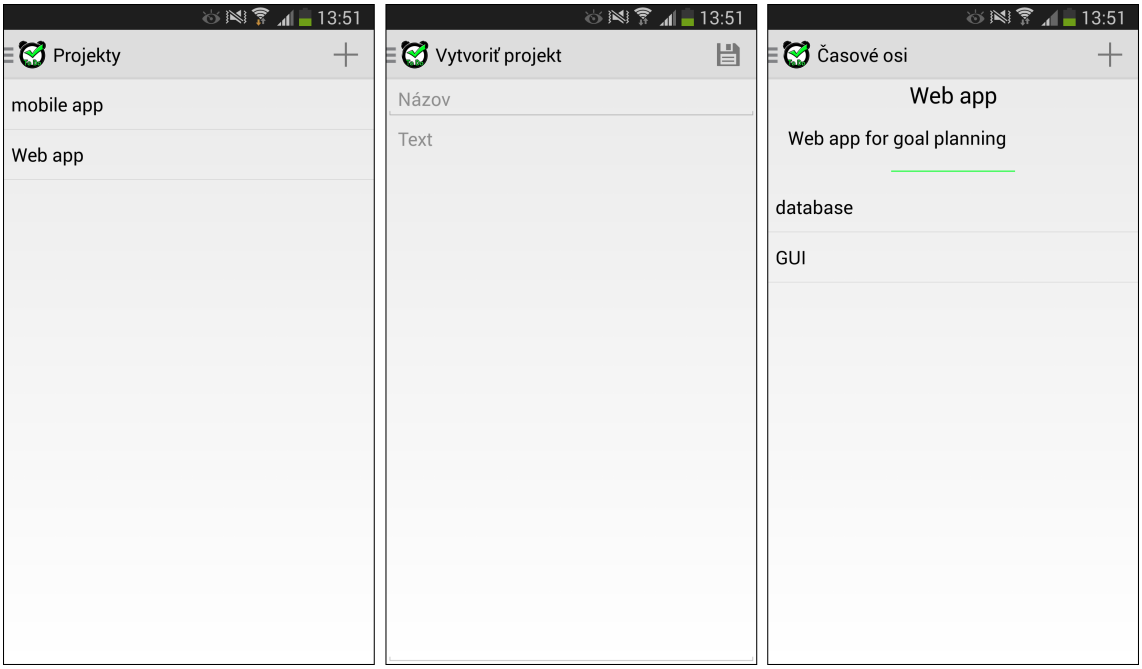
Fragment odvodený od triedy ListFragment. Zobrazuje projekty a poskytuje ich správu. Pri podržaní položky jedného z projektov sa zobrazí kontextové menu s voľbami upraviť a odstrániť. Pri vytváraní sa v prekrývajúcej metóde `onCreateView` pripojí k rozloženiu LinearLayout obsahujúci ListView. Je v ňom implementovaná metóda `updateList`, ktorá je volaná pri vytváraní fragmentu v metóde `onActivityCreated`. updateList slúži na vytvorenie adaptéra a jeho následné nastavenie prvku ListView.

Ako adaptér je použitá preddefinovaná trieda SimpleCursorAdapter. Jej konštruktor je volaný s argumentami kontext, identifikátor zdroja, kde sa nachádza XML súbor s rozložením, kurzor do databázy, pole reťazcov označujúce stĺpce tabuľky v databáze, z ktorých budú použité dáta, pole identifikátorov grafických prvkov, do ktorých budú dáta vložené, číselnú hodnotu príznaku. Databázový kurzor je získaný dotazom (metóda `query`) triedy ContentResolver, ktorý komunikuje s triedou ContentProvider aplikácie. Metóda onListItemClick je prekrytá a pomocou verejného statického rozhrania `OnProjectClickedListener` je v nej volaná metóda `onProjectClicked`, ktorej parameter je identifikátor projektu. `onProjectClicked` je implementovaná v aktivite PojectActivity.

V prepísanej metóde onCreateContextMenu sa pridajú do kontextového menu možnosti pre upravenie a odstránenie projektu. Obsluha zvolenej voľby menu je implementovaná v metóde `onContextItemSelected`, v ktorej sa získa identifikátor zvolenej položky menu. V prepínači sa podľa tohto identifikátora vyberie obsluha. Buď je volaná metóda `editNote` pre editovanie projektu alebo `deleteNote` pre odstránenie projektu. V oboch prípadoch sa vracia kladná pravdivostná hodnota `pravda`. Ak neboli tieto možnosti identifikované, vracia sa volanie prekrytej metódy `onContextItemSelected`.

Argumentom metód `editNote` a `deleteNote` je identifikátor projektu, ktorý sa získa pomocou triedy AdapterContextMenuInfo. Tá poskytuje po prevedení metódy `getMenuInfo` informácie spojené s vybratou položkou menu. Pridanie viacerých fragmentov typu ListFragment so samostatnými menu ponukami malo za dôsledok, že sa vykonala nesprávna obsluha zvolenej položky menu. Odstránenie tohto problému realizuje podmienka na začiatku metódy `onContextItemSelected`, ktorá overuje, či je fragment viditeľný. Ak nie je, vracia pravdivostnú hodnotu `nepravda`. Metóda `deleteNote` pomocou triedy ContentResolver

zavolá metódu `delete`, ktorej predá odkaz URI na projekt označený na odstránenie. Následne sa zobrazí správa typu Toast oznamujúca používateľovi, či bolo odstránenie vykonané. V metóde `editNote` sa najskôr vytvorí objekt triedy Bundle, do ktorého sa vložia hodnoty novému fragmentu na úpravu projektu. Do neho sa vloží identifikátor projektu. Pomocou dotazu triedy ContentResolver sa získa databázový kurzor ukazujúci na konkrétny projekt. Cez ten sa pristúpi k hodnotám projektu, ktoré sa následne vložia do objektu Bundle.



Obrázek 4.1: Aplikácia pri zobrazení projektov, vytvárania nového projektu, zobrazenie konkrétneho projektu.

Metódou `getFragmentManager` sa sprístupní objekt triedy FragmentManger, na ktorom sa zavolá metóda `beginTransaction`. Táto vráti objekt triedy FragmentTransaction, ktorý sprístupní aplikačné rozhranie pre operácie s fragmentmi. Vytvorí sa fragment AddProjectFragment, ktorému sú odovzdané hodnoty projektu pomocou metódy `setArguments`. Táto metóda je volaná s objektom triedy Bundle ako argumentom. Následne nad objektom triedy FragmentTransaction zavoláme metódu `replace`, ktorá nahradí grafický prvok typu FrameLayout na to určený, za nový fragment. Operácia sa potvrdí metódou `commit`.

4.1.2 Vytvorenie a upravenie projektu – AddProjectFragment

Tento fragment je využívaný pri pridávaní nových, ale aj pri upravovaní už existujúcich projektov. Obidve akcie totiž využívajú rovnaké grafické rozmiestnenie. V metóde `onCreateView` sa okrem vloženia rozloženia overuje, či fragment obsahuje nejaké argumenty. Ak obsahuje hodnoty pre meno a popis projektu, vyplní príslušné grafické prvky EditText. Okrem toho priradí metódu `onSubmitClicked`, ktorá sa zavolá po stlačení tlačidla. V tejto metóde sa spracujú hodnoty zadané používateľom. Ak fragment obsahuje identifikátor projektu, zavolá sa metóda `onEditProject` pre aktualizovanie existujúceho projektu, inak sa vytvára projekt nový metódou `onAddProject`. Tieto metódy sú implementované v aktivite ProjectsActivity a pristupuje sa k nim cez verejné statické rozhranie `OnAddProjectListener`.

4.1.3 Zobrazenie projektu – SingleProjectFragment

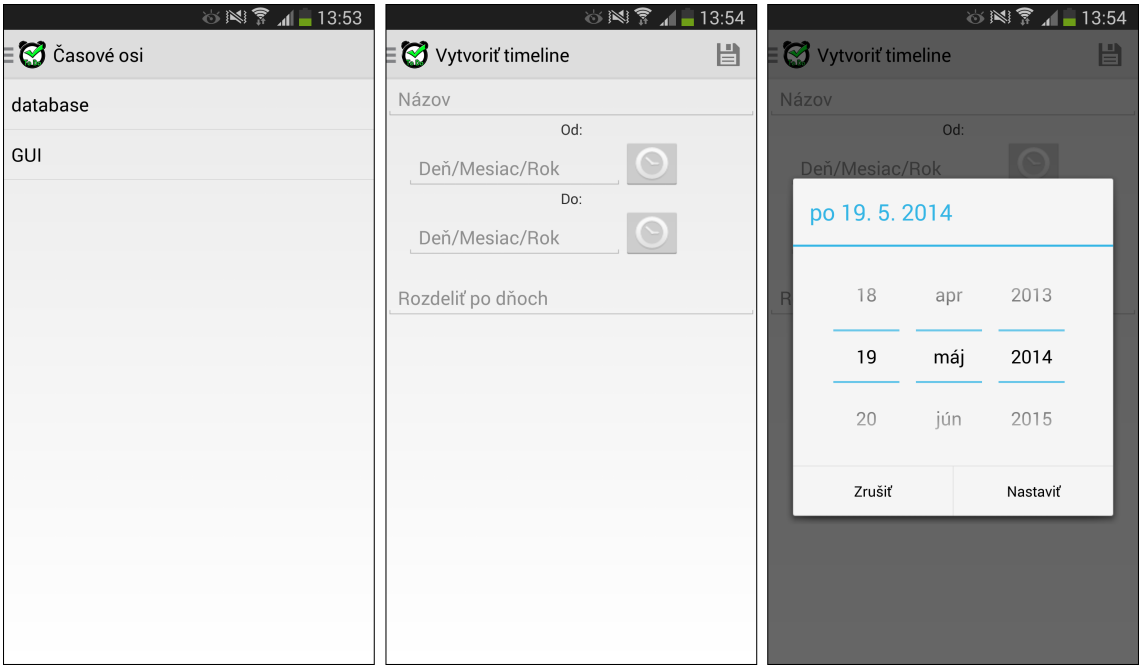
Fragment zobrazujúci jeden projekt a časové osi k nemu priradené. V metóde `onCreateView` je priradené grafické rozmiestnenie a je v nej vytvorený fragment TimelinesListFragment. Tomuto fragmentu je pomocou objektu triedy Bundle odovzdaná hodnota identifikátora projektu. Fragment sa nahradí za grafický prvok FrameLayout, ktorý je na to určený. Grafické prvky TextView popisujúce projekt, sú vyplnené dátami. Tieto sú sprístupnené cez databázový kurzor, ktorý vráti dotaz nad objektom triedy ContentResolver. V dotaze je použitá URI adresa, ktorá bola ako parameter konštruktora triedy SingleProjectFragment uložená do súkromnej členskej premennej.

4.2 Časové osi

V tejto časti sú popísané triedy implementujúce zobrazovanie a správu časových osí.

4.2.1 Zoznam časových osí – TimelinesListFragment

Fragment zobrazuje a spravuje časové osi. Je podobný triede ProjectsListFragment, s tým rozdielom, že obsluha akcií a grafické rozmiestnenie je prispôsobené pre časové osi. Menu vyzerá rovnako. Metóda `updateList` sa líši najmä v tom, že pokiaľ fragment obsahuje argument s hodnotou identifikátora projektu, sú zobrazené iba časové osi tohto konkrétneho projektu, inak sú zobrazené všetky. Prvku ListView je priradený adaptér triedy SimpleCursorAdapter.



Obrázek 4.2: Zoznam časových osí, vytváranie osi, výber dátumu.

Metóda `editTimeline` volaná po vybraní položky kontextového menu získa dáta časovej osi, ktorá bola zvolená. Tie vloží do objektu triedy Bundle, ktorý vloží do fragmentu AddTimelineFragment. Pri vkladaní dátumov pre určenie trvania časovej osi sú dáta prevedené zo sekúnd od 1. 1. 1970 do reťazcov tvaru `deň/mesiac/rok`. Konverzia je realizovaná

objektom triedy Calendar, ktorý je nastavený metódou `setTimeInMillis` na konkrétny dátum hodnotou z tabuľky databázy násobenou hodnotou tisíc. Pre získanie dňa, mesiaca, roku dátumu, je volaná nad objektom triedy Calendar metóda `get`, ktorej je ako argument predaná konštanta políčka `Calendar.DATE`, `Calendar.MONTH`, `Calendar.YEAR`. Vrátená hodnota mesiaca je inkrementovaná o jedna z dôvodu, že v metóde `get` triedy Calendar sú mesiace číslované od nuly.

4.2.2 Vytvorenie a upravenie časovej osi – AddTimelineFragment

Fragment pre vytváranie a upravovanie časových osí. Funkčne je podobný triede AddProjectFragment, no pribudli nové grafické prvky. Používateľ musí vyplniť obdobie trvania časovej osi. Dátumy začiatku a konca je možné určiť textom vo formáte `deň/mesiac/rok` alebo stlačením tlačidla s obrázkom hodín. Po stlačení tlačidla je spustená metóda `selectDate`. Tá je implementovaná v aktivite ProjectActivity. Vytvára nový fragment SelectDateFragment, do ktorého vloží hodnotu identifikátora pohľadu. Ten získa pomocou metódy `getId` triedy View. Pre zobrazenie dialógu fragmentu SelectDateFragment sa použije metóda `show`.

Trieda SelectDateFragment je implementovaná priamo v ProjectActivity, pretože nie je nikde inde využívaná a slúži iba na výber a vloženie dátumu do príslušných políčok. Je odvodená od triedy DialogFragment a implementuje rozhranie `OnDateSetListener` triedy DatePickerDialog. Pri vytváraní fragmentu je spustená metóda `onCreateDialog`. V nej sa inicializuje objekt triedy Calendar pomocou metódy `getInstance`. Tým bude objekt nastavený na aktuálny čas podľa systému a prednastavenej časovej zóny systému. Metóda `onCreateDialog` vracia nový objekt triedy DatePickerDialog, ktorý nastaví na aktuálny čas pomocou objektu triedy Calendar. Metóda `onDateSet` rozhrania `OnDateSetListener` zavolá metódu aktivity ProjectActivity – `populateSetDate`, ktorú zavolá s argumentmi rok, mesiac, deň nastavenými v dialógu a identifikátorom pohľadu. V metóde `populateSetDate` sa podľa identifikátora pohľadu rozhodne, či sa bude vkladať text do políčka pre začiatok alebo koniec časovej osi. Následne sa vloží do políčka text vo formáte `deň/mesiac/rok`.

Metóda `onSubmitClicked` triedy AddTimelineFragment používa na spracovanie zadaného textového formátu dátumu objekt triedy SimpleDateFormat. Jeho konštruktor prijíma ako argument reťazec šablóny, ktorý určuje formátovanie dátumu. Reťazec určujúci šablónu je `\dd/MM/yyyy\`.[1] Objekt triedy SimpleDateFormat sa pokúsi spracovať reťazec z grafického prvku EditText a vytvoriť z neho objekt triedy Date reprezentujúci dátum. Z tohto objektu sa metódou `getTime` získa hodnota milisekúnd od 1. 1. 1970. Táto hodnota je delená hodnotou tisíc, pretože nám stačí hodnota v sekundách. Takto upravené hodnoty začiatku a konca časovej osi sú spolu s názvom a veľkosťou časových úsekov argumentami medódy `onAddTimeline`. Veľkosť časových úsekov je načítaná z políčka EditText, ktorý umožňuje zadať iba číselnú hodnotu. Zadaná hodnota reprezentuje počet dní časového úseku. Časová os bude rozdelená na časové úseky, aby mal používateľ lepší prehľad o termínoch svojich úloh.

Metóda `onEditTimeline` má parametre názov a identifikátor časovej osi. Po vytvorení časovej osi nie je možné meniť jej časové ohraničenie a veľkosť časových úsekov. Príčinou je nejednoznačné určenie priradenia úloh k časovým úsekom, ktorých počet a obdobie by sa mohli zmeniť. Prístup k metódam aktivity ProjectActivity je realizovaný cez rozhranie `OnAddTimelineListener`.

[1] d – deň v mesiaci, M – mesiac v roku, y – rok

4.2.3 Zobrazenie časovej osi – SingleTimelineFragment

Fragment, ktorý zobrazuje jednu časovú os a k nej patriace časové úseky, ktoré obsahujú úlohy. Jeho konštruktor má parameter URI adresu časovej osi, ktorá bude zobrazená. Pomocou URI adresy získa objekt triedy ContentResolver dotazom kurzor, ukazujúci na konkrétny záznam časovej osi v tabuľke časových osí databázy. Kurzor nastavený na prvý riadok metódou `moveToNext`, získa metódou `getString` názov časovej osi. Metóda `getString` je volaná s argumentom určujúcim index stĺpca, v ktorom sa hodnota nachádza. Časové úseky a úlohy sú zobrazené vo fragmente EventsListFragment, ktorý je vložený pod prvok TextView s názvom časovej osi. Vloženie je realizované v XML súbore s grafickým rozmiestnením, ktoré bolo pridané pri vytváraní fragmentu v metóde `onCreateView`. V tomto súbore je vloženie deklarované značkou include s atribútom layout. Hodnotou atribútu je odkaz na XML súbor s rozmiestnením, v ktorom je deklarovaný fragment EventsListFragment.

4.3 Úlohy

V tejto časti sú popísané triedy implementujúce zobrazovanie a správu úloh.

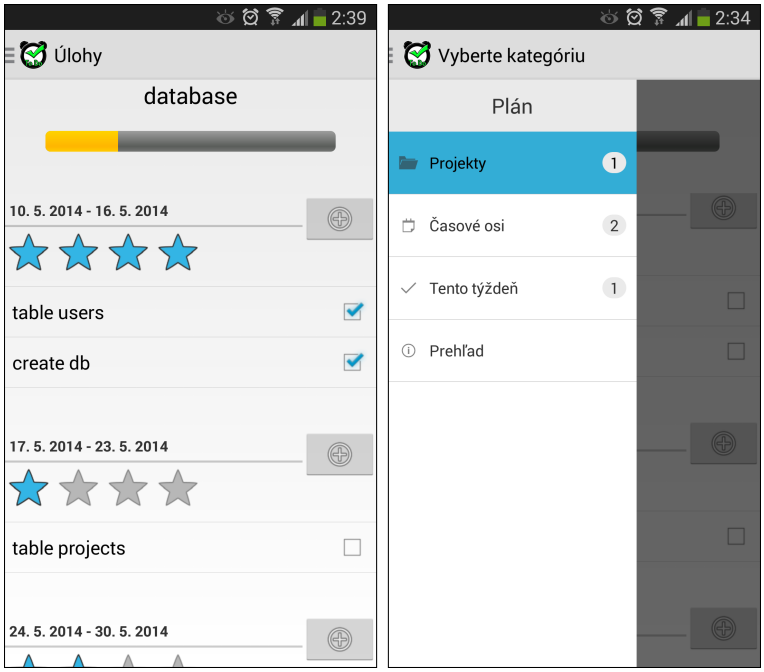
4.3.1 Zoznam úloh – EventsListFragment

Tento fragment zobrazuje časové úseky časovej osi, ktorá bola vybraná. Každý časový úsek má hlavičku. Tá sa skladá z prvkov TextView, RatingBar, a ImageButton. Do prvku TextView sa nastaví text dátumov trvania úseku. TextView má priradený štýl `listSeparatorTextViewStyle`, aby boli úseky jasne graficky oddelené. Prvok RatingBar slúži používateľovi na hodnotenie spokojnosti s plnením úloh v danom časovom úseku. Prvok ImageButton umožňuje pridávať úlohy k príslušnému časovému úseku. Časový úsek môže obsahovať úlohy. Úlohy sú interpretované prvkami CheckedTextView, aby bolo možné zvýrazniť splnené úlohy.

Trieda adaptéra, ktorý by umožňoval zanoriť do seba viac kurzorov a vložiť ich dáta do prvku ListView v aplikačnom rozhraní systému Android nie je. Adaptér a prvok ListView sú však pre tento účel vhodné. Vyriešil som to implementovaním vlastnej triedy CustomCursorAdapter, ktorá je odvodená od triedy BaseAdapter. Táto trieda spája adaptéry odvodené od triedy CursorAdapter. Pre hlavičky časových úsekov je použitá vlastná trieda HeaderCursorAdapter. Každá hlavička má svoj vlastný adaptér triedy SimpleCursorAdapter pre úlohy v danom úseku. V konštruktore triedy `CustomCursorAdapter` sú najskôr inicializované hodnoty potrebné pre vytvorenie adaptérov. Hodnota identifikátora časovej osi sa získa z argumentu tohto fragmentu, ktorý bol do neho vložený pri vytváraní.

Adaptér HeaderCursorAdapter je vytvorený s kurzorom, ktorý vybral všetky časové úseky patriace k otvorenej časovej osi. Pomocou metódy `getCount`, zavolanej nad objektom triedy Cursor, sa určí počet časových úsekov. Vytvorí sa pole adaptérov triedy SimpleCursorAdapter o veľkosti danej počtom. V cykle s počtom opakovaní podľa množstva časových úsekov sa v každej iterácii prejde na ďalší riadok kurzora časových úsekov a do príslušnej položky poľa sa vloží nový adaptér triedy SimpleCursorAdapter. Ten je vytvorený kurzorom, ktorý pomocou identifikátora úseku vyberie úlohy, ktoré patria k danému časovému úseku. V každej iterácii sa okrem toho ukladajú pozície hlavičiek a pripočítava sa počet položiek do celkového súčtu položiek adaptéra CustomCursorAdapter.

Preto, aby boli splnené úlohy podľa hodnoty z databázy označené, bolo nutné zmeniť plnenie dát adaptérom do grafických prvkov. To bolo dosiahnuté zmenením objektu triedy



Obrázek 4.3: Zobrazenie úloh časovej osi, výber menu.

ViewBinder, objektu adaptéra SimpleCursorAdapter, metódou `setViewBinder`, ktorej argumentom je vlastný ViewBinder, `eventViewBinder`. `eventViewBinder` obsahuje vlastnú implementáciu metódy `setViewValue`, ktorá prekrýva pôvodnú. Získa z kurzora reťazec a príznak splnenia úlohy. Reťazec je nastavený ako text prvku CheckedTextView metódou `setText` a ak príznak určuje splnenú úlohu, je CheckedTextView označený metódou `setChecked`.

Pre správnu funkčnosť adaptéru CustomCursorAdapter bolo nutné implementovať nasledujúce metódy. Metóda `getCount` vráti počet položiek. Keďže každý časový úsek má jednu hlavičku a adaptér s úlohami, súčet je získaný iterovaním cez všetky adaptéry. Do súčtu sa v každej iterácii pripočíta počet položiek adaptéra úloh pomocou metódy `getCount` a jedna hlavička úseku. Metóda `getViewTypeCount` vracia počet typov grafických prvkov použitých v adaptéri. V tomto prípade sú dva typy, jeden pre hlavičky a druhý pre úlohy. Metóda `getItemViewType` vracia číselnú hodnotu určujúcu typ grafického prvku a jej parameter je pozícia položky adaptéra. V cykle sa prejdú všetky pozície hlavičiek časových úsekov, ktoré boli v konštruktore uložené. V každej iterácii sa porovná pozícia s pozíciou hlavičky. Ak sú rovnaké, vráti konštantu `TYPE_SECTION_HEADER`. Ak je pozícia menšia ako pozícia hlavičky, znamená to, že položka nebude hlavičkou a vráti sa konštanta `TYPE_SECTION_EVENT`. Metóda `getItem` vracia dáta na pozícii danej parametrom. Cyklus iteruje cez adaptéry úloh. V každej iterácii cyklu sa uloží súčet položiek časového úseku. Ten určuje počet úloh plus jedna hlavička. Ak je pozícia rovná nule, je položka hlavičkou. V tom prípade sa nad adaptérom hlavičiek zavolá metóda `getItem`, ktorej argumentom je index časového úseku. Ak je pozícia menšia ako súčet položiek úseku, je to úloha. Vtedy sa pristúpi k adaptéru indexom úseku do poľa adaptérov úloh, nad ktorým sa zavolá metóda `getItem`. Jej argumentom bude pozícia zmenšená o jednotku, pretože musíme odpočítať hlavičku. V obidvoch prípadoch je návratová hodnota určená výsledkom volania metódy `getItem`. Ak ani jedna z týchto dvoch podmienok nebude pravdivá, od pozície sa odpočíta

súčet položiek časového úseku a cyklus pokračuje ďalej. Metóda `getView` vracia grafický prvok položky určenej parametrom. Je v nej podobný cyklus ako v metóde `getItem`. Namiesto `getItem` sa však použije metóda `getView`. Ak ide o úlohu, overí sa, či by mala byť úloha označená a prípadne zavolá nad objektom triedy ListView metódu `setItemChecked`. Metóda `getItemId` obsahuje rovnaký cyklus, ale vracia výsledky volaní metód `getItemId`. Metóda slúži na zistenie identifikátora položky zoznamu.

Trieda HeaderCursorAdapter je odvodená od triedy CursorAdapter a implementuje rozhranie `OnRatingBarChangeListener`. Metóda rozhrania `onRatingChanged` je volaná pri zmene atribútu rating prvku RatingBar. Rozlíšenie volania spôsobené menením hodnoty vkladanej z databázy, od zmeny hodnoty používateľom, je realizované pravdivostnou hodnotou parametra fromUser. Ak bola zmena prevedená používateľom, hodnota v databáze sa aktualizuje pomocou objektu triedy ContentResolver. Identifikácia časového úseku, ku ktorému patrí hodnotenie, je vrátená volaním metódy `getTag` nad prvkom RatingBar. Metóda `newView` vytvára nový grafický prvok triedy View, zastupujúci jednu položku v zozname, do ktorého budú z kurzora vkladané dáta. Položku zoznamu HeaderCursorAdapter reprezentujú prvky TextView pre názov úseku, RatingBar pre ohodnotenie spokojnosti s plnením úloh v danom úseku, ImageButton pre pridávanie úloh do úseku. Rozhranie implementované v tejto triede je položke RatingBar v tejto metóde nastavené metódou `setOnRatingBarChangeListener`. V metóde `bindView` sa hodnoty z kurzoru vkladajú do grafických prvkov. Prvkom ImageButton a RatingBar je metódou `setTag` nastavená hodnota identifikátora časového úseku, ku ktorým prvky patria.

Metóda `onListItemClick` je volaná v prípade označenia niektorej úlohy. Nad parametrom metódy typu View pretypovaným na typ CheckedTextView, je volaná metóda `setChecked`, ktorá vráti pravdivostnú hodnotu reprezentujúcu označenie položky. Pomocou objektu triedy ContentResolver sa aktualizuje hodnota v databáze. Ak bola položka označená, aktualizovanou hodnotou bude nula, inak sa nastaví jedna.

Trieda EventsListFragment má podobné kontextové menu, ktoré sa zobrazí po stlačení a podržaní položky zoznamu úloh. Pri zvolení voľby menu sa zavolá príslušná metóda `editEvent` alebo `deleteEvent`. Úprava a odstraňovanie úloh je riešená podobne ako v triedach ProjectsListFragment, TimelinesListFragment. Pri úprave sa teda vytvorí fragment pre vytvorenie a editovanie úlohy AddEventFragment, do ktorého sa vložia hodnoty upravovanej úlohy.

4.3.2 Úlohy v aktuálnom týždni – CurrentEventsListFragment

Úlohy je možné vidieť v zobrazení časovej osi, ku ktorej patria. Fragment CurrentEventsListFragment zobrazuje zoznam úloh všetkých osí, ktoré patria k aktuálnemu týždňu. Poskytuje tak používateľovi prehľad aktuálnych úloh, ktoré má splniť. Na vrchu je dátum začiatku a konca týždňa. Pod ním sa nachádza zoznam úloh.

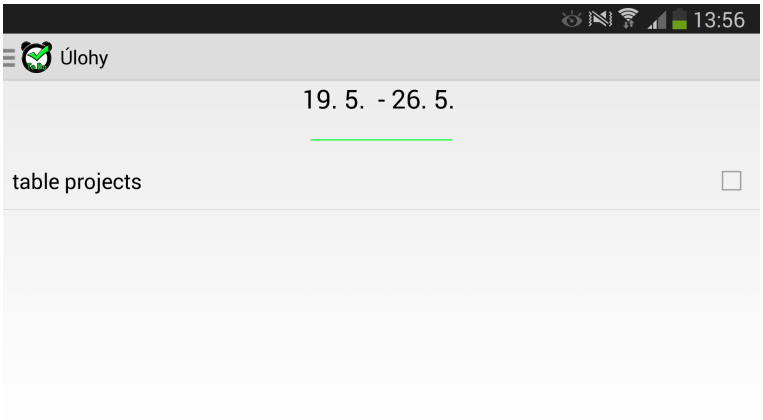
Statická metóda `CursorsOfWeek` má parametre objekt triedy Context, pomocou ktorého pristupuje metóda k objektu triedy ContentResolver a grafický prvok TextView. Pomocou triedy Calendar sa najskôr metódou `getInstance` získa inštancia s nastaveným aktuálnym časom. Tá sa pomocou metód `set` a `clear`, volanými s príslušnými konštantami triedy Calendar, nastaví na začiatok aktuálneho týždňa. Metóda `getTimeInMillis` vráti čas v milisekundách, ktorý po delení konštantou tisíc budeme porovnávať s hodnotami v databáze. Nastavenie času inštancie na koniec týždňa zabezpečuje metóda add volaná s argumentmi konštantou určujúcu pridanie týždňa a číslo určujúce počet týždňov, ktoré budú pridané (v tomto prípade jeden). Ak parameter prvku TextView nemá hodnotu `null`, nastaví sa

prvku text s dátumami, určujúci aktuálny týždeň. Pre správne časové ohraničenie je použitý dotaz cez objekt triedy ContentResolver, určujúci tabuľku časových úsekov s podmienkou pre výber podľa času. Podmienka vyberie úseky s dátumom konca väčším alebo rovným ako začiatok týždňa a súčasne s dátumom začiatku menším alebo rovným ako koniec týždňa. Cyklus prechádza kurzorom s časovými úsekmi a vyberá identifikátor. Pomocou identifikátora sa v dotaze vyberú úlohy patriace k úseku. Kurzor úloh je vložený do poľa kurzorov. Po poslednej iterácii sa vytvorí objekt triedy MergeCursor konštruktorom s polom kurzorov úloh ako argumentom. Tento objekt metóda vracia ako výsledok.

V metóde `updateList` sa volá metóda `CursorsOfWeek` s objektom triedy Context a prvkom TextView pre nastavenie nápisu. Vrátený kurzor triedy MergeCursor sprostredkuje spojenie kurzorov úloh rôznych časových úsekov. Adaptér triedy SimpleCursorAdapter je vytvorený s kurzorom a grafickým rozložením určujúcim grafické prvky CheckedTextView, použité pri vytvorení zoznamu. Adaptéru je nastavený metódou `setViewBinder` objekt triedy ViewBinder `eventViewBinder` ako vo fragmente EventsListFragment.

Metóda `onListItemClick` zabezpečuje aktualizovanie hodnoty pre indikáciu splnenia úlohy v databáze a aktualizovanie zoznamu úloh.

Fragment má kontextové menu s položkami pre upravenie a odstránenie úlohy ako fragment EventsListFragment. Metódy `editEvent` a `deleteEvent` sa starajú o obsluhu týchto možností.



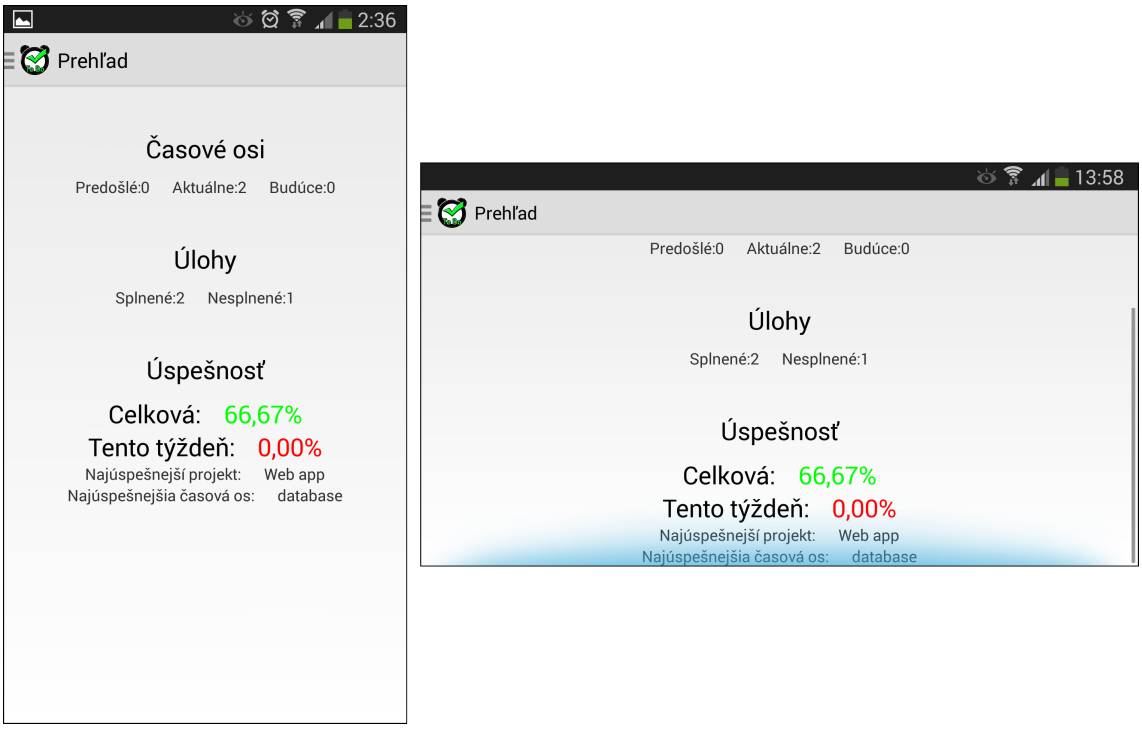
Obrázek 4.4: Zobrazenie úloh aktuálneho týždňa.

4.4 Prehľad plánovania – StatsFragment

StatsFragment zobrazuje prehľad plánovania používateľa. Súčty riadkov výberu v kurzore vracia metóda `getCount`. Zobrazuje počet minulých, aktuálnych a budúcich časových osí. Počty sú získané podmienkou v dotaze porovnávajúcou aktuálny čas s časom začiatku a konca časových osí. Pod časovými osami sú zobrazené súčty splnených a nesplnených úloh, ktoré sú získané selekciou podľa hodnoty stĺpca tabuľky done. Pod súčtami úloh sú zobrazené údaje o úspešnosti. Celková úspešnosť je daná pomerom splnených a nesplnených úloh. Úspešnosť aktuálneho týždňa udávajú úlohy v ňom riešené. Kurzor na tieto úlohy vráti metóda `CursorOfWeek`. Cyklus prechádza kurzorom a v prípade, že je úloha splnená navyšuje hodnotu premennej určujúcu počet splnených úloh. Najúspešnejší projekt a najúspešnejšia časová os sa zistí prejdením všetkých projektov, časových osí, úsekov

a úloh realizovaným vnorenými cyklami. V každej iterácii cyklu, prechádzajúceho projekty, sa pred koncom iterácie vyhodnotí úspešnosť úloh. Ak je úspešnosť lepšia ako v premennej `projectSuccess`, ktorá je prednastavená na hodnotu nula, je úspešnosť do tejto premennej vložená a do premennej `projectName` sa nastaví meno projektu, ktorý je v tejto iterácii prechádzaný. Tak isto sa porovnáva úspešnosť iterácií cez časové osi. Úspešnosť sa nastavuje do premenných `timelineSuccess` a `timelineName`. Všetky hodnoty sú následne metódou `setText` nastavené príslušným prvkom TextView. Prvkom TextView s hodnotami úspešnosti sú podľa úspešnosti zmenené farby textu metódou `colorOfSuccess`. Grafické prvky sú umiestnené v prvku ScrollView. To zabezpečuje korektné zobrazenie dát na menšom displeji alebo po pretočení zariadenia.

Parametre metódy `colorOfSuccess` sú hodnota typu `float` a prvok TextView. Metóda nastavuje prvku TextView farbu podľa hodnoty parametru vyjadrujúceho úspešnosť plnenia úloh. Pre rozdelenie na tri úrovne úspechu sa používajú konštanty `DONW` a `UP` reprezentujúce spodnú a hornú hranicu. Metódou `setTextColor` sa zmení farba textu prvku TextView.



Obrázek 4.5: Zobrazenie prehľadu postupu v pláne pred a po zmene orientácie.

4.5 Aktivita aplikácie – ProjectActivity

Trieda ProjectActivity je jedinou aktivitou aplikácie. Je odvodená od triedy ActionBarActivity, aby aplikácia mohla používať grafický prvok ActionBar. Implementuje metódy rozhraní fragmentov. Tieto metódy slúžia pre obsluhu zobrazenia, ukladania nových a upravených projektov, časových osí, úloh. Pri vytvorení triedy je vytvorené pole s identifikátormi obrázkov, použitých ako ikony položiek menu a pomocné konštanty pre prístup do hašovacej tabuľky.

V metóde `onCreate` sa metódou `setContentView` nastaví grafické rozloženie DrawerLa-

yout. Nastavia sa hodnoty pre prvky ActionBar a hlavné menu. Pre hlavné menu je použitý prvok DrawerLayout. Názvy položiek menu sú uložené ako pole reťazcov medzi zdrojmi v súbore s reťazcami `strings.xml`. Prístup k nemu je možný metódami `getResources` a `getStringArray`. Vytvorí sa zoznam objektov triedy HashMap. V cykle s počtom opakovaní podľa počtu položiek menu, sa budú vytvárať hašovacie tabuľky pre položky menu a pridávať do zoznamu. Do tabuliek budú vložené názvy, identifikátory obrázkov položiek a súčty prvkov patriacim k položkám menu. Zoznam je vložený do adaptéra triedy SimpleAdapter. Parametre konštruktora adaptéra určujú, do akých prvkov a ktorého grafického rozloženia budú položky prvkov zoznamu vkladané. Pre prepojenie prvkov ActionBar a DrawerLayout sa vytvorí objekt triedy ActionBarDrawerToggle, ktorý vytvorí tlačidlo pre zobrazenie menu a rozhranie. V rozhraní sú implementované metódy pre obsluhu po otvorení menu `onDrawerOpened` a po zatvorení `onDrawerClosed`. Po otvorení je volaná metóda `setCounts` pre nastavenie súčtov prvkov položiek a nastaví sa názov prvku ActionBar. Po zatvorení sa volá metóda pre aktualizovanie názvu prvku ActionBar. V obidvoch prípadoch sa volá metóda `supportInvalidateOptionsMenu`, ktorá signalizuje zmenu menu. Následne sa rozhranie priradí prvku DrawerLayout a prvku ListView reprezentujúcemu zoznam menu položiek sa nastaví adaptér. Obsluha výberu položky menu zavolá metódu `showFragment`.

Metóda `showFragment` podľa parametra, ktorým je pozícia položky menu, nastaví prvku ActionBar názov príslušný k zvolenej kategórii a vytvorí príslušný fragment pre zobrazenie zvolenej kategórie. Ten následne zobrazí. Metóda `setSelectedItem` získa metódou `getCheckedItemPosition`, volanou nad zoznamom položiek menu, pozíciu zvolenej kategórie. Ak je pozícia správna, nastaví prvku ActionBar názov podľa pozície.

Metóda `getCountOfItems` vracia počet prvkov v kategórii určenej parametrom. Nad kurzorom, získaným dotazom nad objektom triedy ContentResolver, je zavolaná metóda `getCount`. Metóda `setCounts` prechádza menu zoznam a každej položke nastaví súčet jej prvkov. Ten získa metódou `getCountOfItems`. V prípade, že je súčet nulový, nastaví sa na riadok hašovacej tabuľky položky menu, určený pre súčet, prázdny reťazec. Inak sa nastaví reťazec so súčtom. Po aktualizovaní dát sa to metódou `notifyDataSetChanged` oznámi adaptéru.

V metóde `onCreateOptionsMenu` sa vytvorí menu prvku ActionBar pridaním XML súboru, kde je definovaný. V tomto menu je viac možností, ale viditeľná je vždy iba časť menu, podľa toho, kde v aplikácii sa používateľ nachádza. Je toho dosiahnuté nastavením viditeľnosti, ktorá je položkám nastavená metódou `setVisible`, ktorej parameter je pravdivostná hodnota. Fragment s odlišným menu prvku ActionBar volá v metóde `onCreateView` metódu `setHasOptionsMenu`, s pravdivostnou hodnotou pravda ako argumentom. Tým sa vyvolá metóda `onCreateOptionsMenu`, v ktorej má fragment nastavené, ktoré položky budú viditeľné.

Metóda `onOptionsItemSelected` obsluhuje stlačenie možnosti v prvku ActionBar. Podľa identifikátora možnosti volá príslušnú akciu. Aktivita obsluhuje možnosti pre kliknutia na vytvorenie nových projektov, časových osí, pre ktoré volá príslušné metódy `onAddProjectClicked`, `onAddTimelineClicked`. Tie vytvoria a zobrazia fragmenty AddProjectFragment, AddTimelineFragment pre vytvorenie nových projektov, časových osí. Fragmenty pre pridávanie a úpravu projektov, časových osí, úloh obsluhujú možnosť pre ich uloženie vo vlastných metódach `onOptionsItemSelected`. Po zvolení uloženia sa volá metóda `onSubmitClicked`. Táto skončí volaním jednej z metód `onAddProject`, `onEditProject`, `onAddTimeline`, `onEditTimeline`, `onAddEvent`, `onEditEvent` podľa kontextu. Tieto metódy používajú objekt triedy ContentResolver pre vkladanie do databázy a aktualizáciu jej hodnôt. Pri upravovaní je volaná metóda `update` s argumentmi pre selekciu podľa identifikátora konkrétnej položky,

na rozdiel od metódy `insert` pri vytváraní nových položiek. Ak je operácia úspešná, vráti sa aplikácia do stavu pred pridávaním/upravovaním metódou `onBackPressed`. Ak úspešná nie je, je o tom používateľ informovaný správou triedy Toast.

V metóde `onAddTimeline` sa do databázy nevkladá iba časová os, ale aj jej časové úseky. Ak vloženie časovej osi prebehne bez chyby, vytvorí sa zoznam objektov triedy ContentValues. O vytváranie objektov a vkladaní do zoznamu sa stará cyklus, ktorý po každej iterácii navyšuje hodnoty dátumov ohraničenia trvania úsekov. Hodnota začiatku sa navýši o jeden deň, preto aby ďalší úsek začínal od nasledujúceho dňa. Hodnota označujúca koniec úseku je rovná súčtu začiatku úseku a počtu dní zvoleného používateľom. Ukončujúcou podmienkou je, ak je hodnota označujúca koniec väčšia ako koniec časovej osi. Ak je potom hodnota začiatku úseku menšia ako koniec časovej osi, prevedie sa korekcia. Zoznam je prevedený na pole, ktoré sa predá metóde `bulkInsert`, zabezpečujúcej hromadné vloženie do databázy.

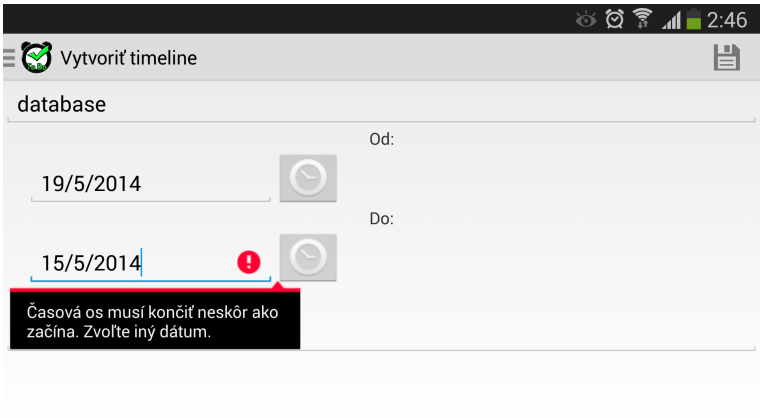
Pre obsluhu akcie po vybratí projektu, časovej osi, slúžia metódy `onProjectClicked`, `onTimelineClicked`, ktoré sú spúšťané z fragmentov. V metódach sa vytvorí fragment pre zobrazenie vybranej položky, ktorý sa následne zobrazí. Zobrazovanie fragmentov je realizované zámenou za prvok FrameLayout v grafickom rozložení, ktorý je na to určený.

4.6 Správca obsahu – TimelineProvider

Trieda TimelineProvider poskytuje rozhranie pre prácu s databázou. Sú v nej definované konštanty pre určenie typu, názvov databázy, tabuliek, s ktorými trieda pracuje. Pre rozlíšenie URI adresy v metódach pre prácu s databázou je použitý objekt triedy UriMatcher. TimelineProvider obsahuje statickú triedu DatabaseHelper, odvodenú od triedy SQLiteOpenHelper. V jej konštruktore sa vytvorí databáza podľa zvoleného mena a verzie. Metóda `onCreate` vykoná SQL príkazy pre vytvorenie tabuliek. V metóde `onCreate` triedy TimelineProvider sa vytvorí objekt triedy DatabaseHelper. V metódach `query`, `insert`, `update`, `delete`, `bulkInsert` sa podľa porovnania parametra s URI adresou zvolí, nad akou tabuľkou sa budú operácie vykonávať. Metóda `delete` pre odstraňovanie položiek z databázy zabezpečuje okrem odstránenia zvolenej položky aj odstránenie položiek patriacej k danej položke. Pri odstraňovaní projektu je zmazaný projekt, z databázy sa vyberú metódou query so selekciou podľa identifikátora projektu časové osi k nemu patriace. Cyklus prejde vybrané osi a hľadá časové úseky k nim patriace. Tak sa pokračuje až k úlohám. Metóda `bulkInsert` slúži na vkladanie časových úsekov. Parametrom metódy je pole objektov triedy ContentValues, ktorého položky sa vložia do databázy.

4.7 Kontrola vstupov

Aby som zabránil neštandardnému chovaniu aplikácie, implementoval som kontrolu používateľských vstupov. Bolo nutné kontrolovať pridávanie a upravovanie projektov, časových osí a úloh. Vo fragmentoch pre pridávanie a upravovanie, sú v metódach `onSubmitClicked` kontrolované nutné podmienky pre úspešné pridanie, upravenie položky. Po každej chybe, ktorá je zachytená, je používateľovi zobrazené, kde spravil chybu a po kliknutí na vstup je vypísaná nápoveda pre správne zadanie vstupu. Toto sprostredkuje metóda `setError` volaná nad prvkom pre vstup, kde bola zaznamenaná chyba. Každá položka musí obsahovať názov. Pri časovej osi sa musí okrem názvu kontrolovať aj formát zadaných dátumov. Overuje sa, či je dátum začiatku osi skôr, ako dátum konca. Hodnota pre počet dní časových úsekov, na ktoré bude os rozdelená, musí byť číslo väčšie ako nula. Pri editovaní časovej osi



Obrázek 4.6: Zobrazenie chybovej hlášky v prípade nesprávneho vstupu.

je používateľovi znemožnené použiť prvky pre nastavenie dátumov a úsekov ich zakázaním metódou `setEnabled` s pravdivostnou hodnotou `nepravda` ako argumentom.

Kapitola 5

Testovanie a publikácia

5.1 Testovanie

Počas celého vývoja bola aplikácia testovaná v emulátore virtuálnym zariadením Android s verziou systému 2.3.3. Po implementovaní prvej verzie grafického používateľského rozhrania a zaistení základnej funkčnosti bola aplikácia vyexportovaná ako balík apk. Tento balík bol odovzdaný používateľom spolu s inštrukciami k inštalácii. Používatelia musia vlastniť zariadenie so systémom Android s minimálnou verziou systému 2.2. V časti zabezpečenie v nastaveniach zariadenia musí byť povolené inštalovanie z neznámych zdrojov. Okrem toho stačí uložiť balík apk do zariadenia a v programe na správu súborov aplikáciu nainštalovať. Väčšina používateľov počas testovania boli študenti s technickým zameraním, takže s inštaláciou problém nemali. Po každej dôležitej zmene aplikácie bola nová verzia aplikácie poslaná používateľom na testovanie. Prioritným pri testovaní bolo odhaliť chyby, zistiť, čo by bolo vhodné pridať, zmeniť a pri každej novej verzii zistiť hodnotenie zmien. V prípade výskytu chýb bolo vo vývoji prvoradé chyby odstrániť a potom znova odoslať opravenú aplikáciu na testovanie. Zmenšilo sa tým riziko zanesenia nových chýb pri oprave. Ak sa pri testovaní chyba neobjavila, pri výbere ďalších zmien sa bral zreteľ na odozvu používateľov. Časť z pripomienok sa podarilo implementovať a otestovať. Neprehľadnosť medzi úlohami bola vyriešená zoskupovaním úloh do časových úsekov. K časovým úsekom boli pridané hodnotenia v podobe hviezdičiek. Zlá ovládateľnosť bola vylepšená pridaním menu. Väčšine testujúcich sa zdalo pridanie ponuky ActionBar ako zjednodušenie ovládania. Tieto zmeny boli hodnotené pozitívne. Negatívne používatelia hodnotili absenciu zdieľania postupu s inými používateľmi cez internet a možnosť zadávania postupu cez počítač. Približne tretine používateľov chýbala možnosť nastavenia upozornení a synchronizácia s kalendárom Google Calendar a inými službami.

5.2 Publikácia v službe Google Play

Každé zariadenie so systémom Android má prednastavene nainštalovanú aplikáciu Obchod Play. V prípade pripojenia k internetu je možné si cez ňu nainštalovať aplikácie.

Po dokončení úplnej verzie aplikácie a jej testoch používateľmi som ju chcel zverejniť. Najskôr som navrhol logo, ktoré som importoval do aplikácie. Vyexportoval som podpísaný balík aplikácie. Vytvoril som si v službe Google Play účet pre vývojára. Odoslal som aplikáciu spolu s ďalšími údajmi ako popis, obrázky aplikácie. Služba poskytuje veľa možností zlepšujúcich a zjednodušujúcich vývoj. Okrem spravovania aplikácie je možné zobraziť si

rôzne štatistiky. Výhodná je možnosť nastavenia alfa a beta testovania.

Pretože môj projekt je nový postačilo mi alfa testovanie. Vytvoril som si zoznam testerov. Pri pridaní zmien do aplikácie, som ju vložil do alfa testovania a tá sa zobrazila v Obchod Play iba testerom. Testerovi sa v prípade pripojenia k internetu a zistenia novej verzie aplikácie, zobrazí ponuka s jej aktualizáciou. Toto poskytuje oveľa rýchlejšie a jednoduchšie testovanie aplikácie.

Kapitola 6

Záver

Cieľom tejto práce bolo vytvoriť mobilnú aplikáciu pre systém Android, umožňujúcu plánovať dosiahnutie cieľov.

Od základnej verzie aplikácie bola aplikácia testovaná študentmi vysokých škôl. Výsledky testovania v podobe spätnej väzby používateľov ovplyvnili ďalší vývoj aplikácie.

Plná verzia aplikácie umožňuje plánovanie dlhodobých projektov. Používateľ si sám môže zvoliť dĺžku časových úsekov, do ktorých bude zoskupovať úlohy vedúce k dosiahnutiu cieľa. Pre väčšie zapojenie používateľa je umožnené hodnotiť spokojnosť s časovým úsekom. Motiváciu k plneniu úloh by mal zvyšovať aj prehľad priebehu plnenia plánu, poskytujúci informácie o úspešnosti.

Plná verzia aplikácie bola sprístupnená v službe Google Play. Aplikácia je dostupná v troch jazykoch a bola testovaná na rôznych mobilných zariadeniach, s rôznymi verziami systému a rozlíšenia. Aplikácia podporuje staršie verzie systému od verzie 2.2 a zmenu orientácie zariadenia. Podporu starších verzií napriek využitiu nových komponentov sa mi podarilo dosiahnuť bez externých knižníc pomocou knižníc Support Library. Tým by malo byť zabezpečené, že aplikácia bude plne funkčná aj po zmenách systému. Cieľ práce považujem za splnený.

Aj napriek publikácii aplikácie TimeLines v službe Google Play si nemyslím, že by aplikácia nahradila komerčné aplikácie a služby pre plánovanie úloh. Potvrdzujú to aj vyjadrenia používateľov. Viacerí však aplikáciu považujú za použiteľnú a užitočnú, čo považujem za úspech.

Najlepšie možnosti využitia a presadenia aplikácie sú podľa mňa v plánovaní diplomových prác. Toto plánovania by sa zlepšilo zapojením vedúcich prác. K tomu by mohol byť implementovaný modul pre rolu vedúceho, ktorý by umožnoval vedúcim sledovanie postupu svojich diplomantov a ďalšie možnosti ako napríklad dohodnutie konzultácií.

Inšpiráciou podľa aplikácií určených na podobný účel a z odozvy používateľov, sa dajú určiť aj ďalsie smery vývoja aplikácie.

Používatelia by mali možnosť zdieľať postup práce na projektoch s ostatnými používateľmi. Vzájomnou motiváciou a porovnávaním postupov v plánoch, by mohli používatelia dosahovať lepšie výsledky. K tomu bude potrebné implementovať serverovú a klientskú časť a navrhnúť komunikáciu medzi nimi. Aj napriek zmenám musí byť zabezpečené správne ukladanie dát v režime offline a po pripojení na internet ich správne spracovanie serverom. Pri prístupe dát bude nutné implementovať autentifikáciu a autorizáciu používateľov.

Ďalším prínosom môže byť vytvorenie webovej aplikácie a prepojenie s ňou. V tomto rozšírení bude nutné navrhnúť riešenie kolízií. Tie by mohli nastať napríklad v prípade aktualizovania hodnoty webovou službou a offline v mobilnej aplikácii súčasne.

Vylepšením môže byť aj zavedenie systému odmeňovania používateľov. Odmeňovanie môže byť realizované oceneniami v podobe získaných obrázkov, reprezentujúcich odznaky za počet vytvorených projektov, osí, splnených úloh, dosiahnutých úspechov.

Literatura

[1] Allen, G.: *Beginning Android 4.* Apress, 2012, 299 s., iSBN 978-1-4302-3984-0.

[2] Galitz, W. O.: *The essential guide to user interface design: an introduction to GUI design principles and techniques. 2nd ed. New York.* Wiley Computer Pub., 2002, 4 s., iSBN 0-471-084646.

[3] Murphy, M. L.: *Beginning Android 2. New York.* Apress, 2010, 24 s., iSBN 978-1-4302-2629-1.

[4] Ujbányai, M.: *Programujeme pro Android. Vyd. 1. Praha.* Grada, 2012, 13 s., iSBN 978-80-247-3995-3.

[5] Ujbányai, M.: *Programujeme pro Android. Vyd. 1. Praha.* Grada, 2012, 37 s., iSBN 978-80-247-3995-3.

[6] Zichermann, G.; Cunningham, C.: *Gamification by design: implementing game mechanics in web and mobile apps. 1st. ed. Sebastopol, Calif.* O'Reilly Media, 2011, xiv s., iSBN 978-1-449-39767-8.

Dodatek A

Obsah CD

Adresárová štruktúra CD:

- `bin` - priečinok obsahuje balík aplikácie TimeLines.apk
- `img` - priečinok s ikonou aplikácie
- `manual` - priečinok obsahujúci návod k inštalácii a používateľskú príručku
- `poster` - plagát prezentujúci výsledky tejto práce (veľkosť A2)
- `src` - zdrojové súbory aplikácie
- `thesis` - zdrojový tvar písomnej správy